NetCDF based data archiving system proposal for the ITER Fast Plant System Control prototype

R. Castro[1], J. Vega[1], M. Ruiz[2], G. De Arcas[2],
E. Barrera[2], J.M. López[2], D. Sanz[2], B. Gonçalves[3],
B. Santos[3], N. Utzel[4], P. Makijarvi[4]

1 Asociación EURATOM/CIEMAT para Fusión. Spain
2 Technical University of Madrid (UPM), Spain
3 Associação EURATOM/IST, IPFN, Portugal
4 ITER Organization, France

Index

- FPSC project
- Archiving requirements
- Archiving solutions
  - NetCDF-4
  - Storage Cluster
- Tests
- Conclusions

FPSC Project

- Objective: To develop a FPSC prototype focused on Data Acquisition for ITER IO
- Two different form factors for the implementation:
  - PXIe based solution (CIEMAT/UPM): *M. Ruiz (05-7)*
  - ATCA based solution (IST): *B. Goncalves (02-4)*
- The “functional requirements” of FPSC prototype:
  - To provide **high rate data acquisition**, preprocessing, archiving and efficient data distribution among the different FPSC software modules.
  - To interface with CODAC and to provide archiving
  - FPSC software based compatible with RHEL and EPICS
  - To use COTS solutions.

Archiving requirements I

- Remote storage over Ethernet
- Reliable (delaying is not important)
- Long pulse
  - Continuous archiving during pulse
  - Huge amount of data
  - Archived data can be read during the pulse
  - FPSC has to be able to Start/Restart in the middle of the pulse
- Scalable (independent of the number of FPSC involved)

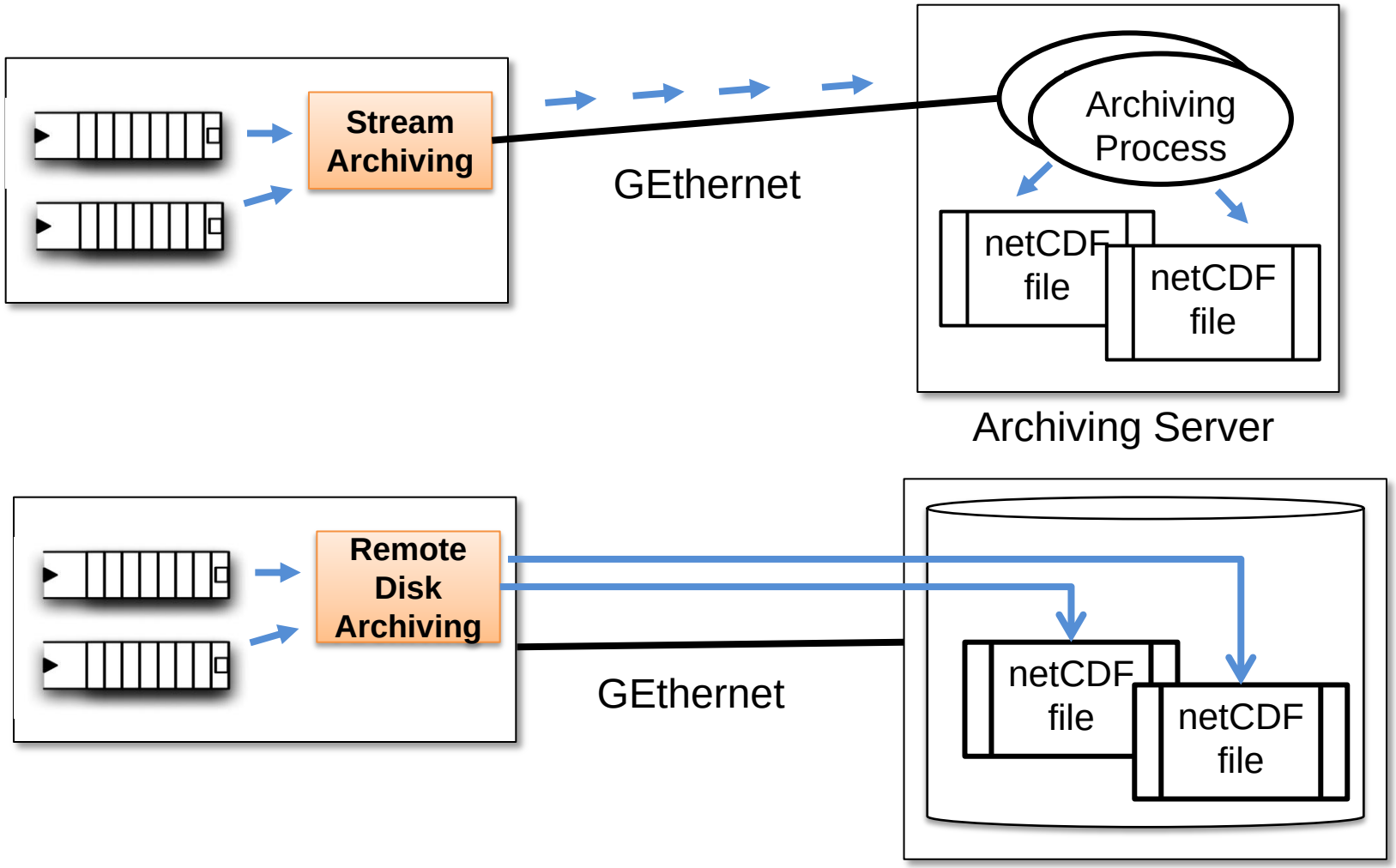
Archiving requirements II

- Valid for different data types
  - Raw acquired signal
  - Processed signal
  - System events
- Integration in EPICS architecture
- Fault Tolerance solution

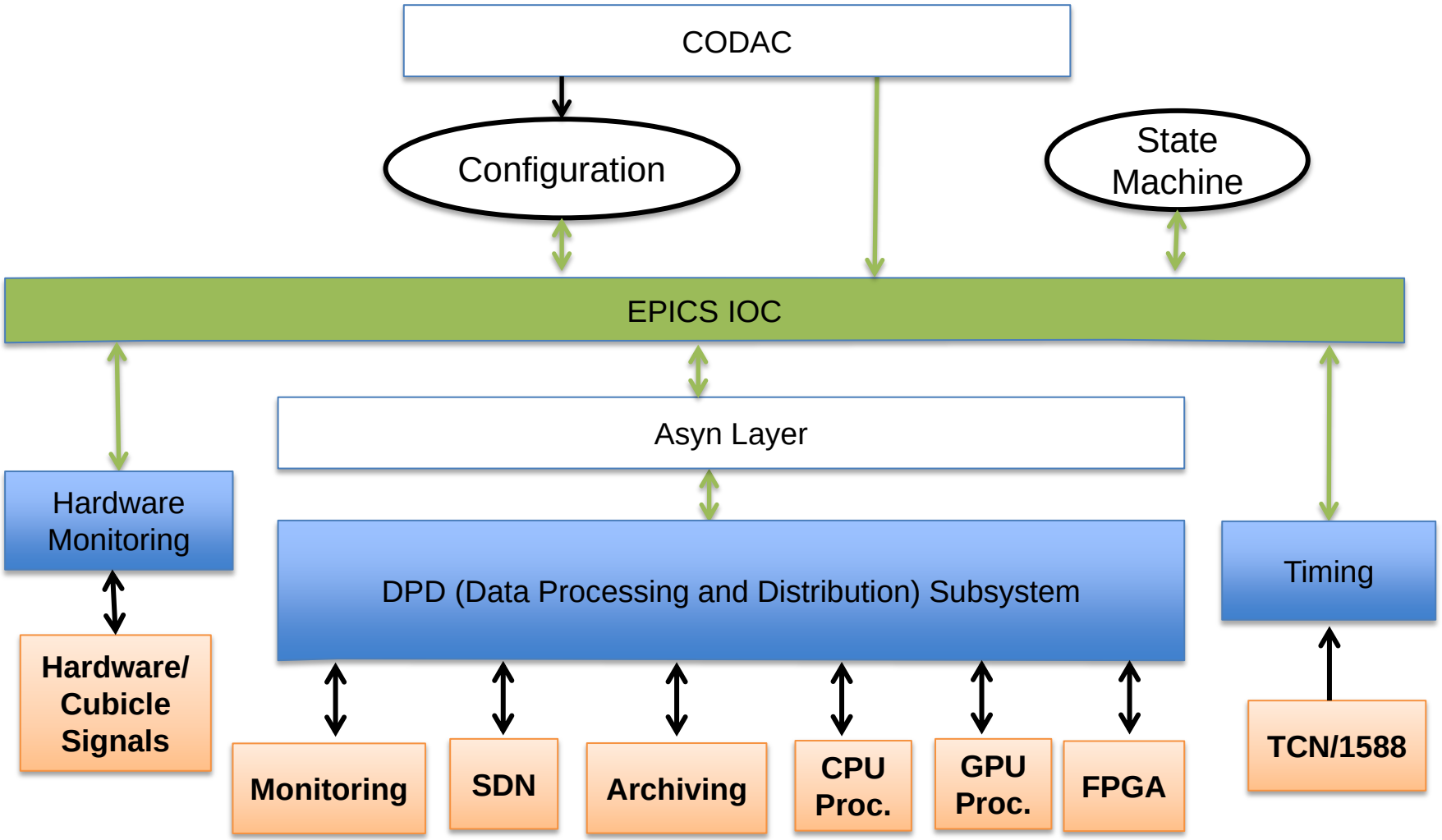
Archiving Solution

- A data source of the FPSC can be assigned to a data archiving service in configuration time
- Storage based on files
- 1 file per data source and pulse
  - Signals: 1 file per signal and pulse
  - System events: 1 file per pulse for all events of a FPSC
- If a data source restarts archiving in the middle of a pulse, data is appended to the existing data file
- Archive format: netCDF-4

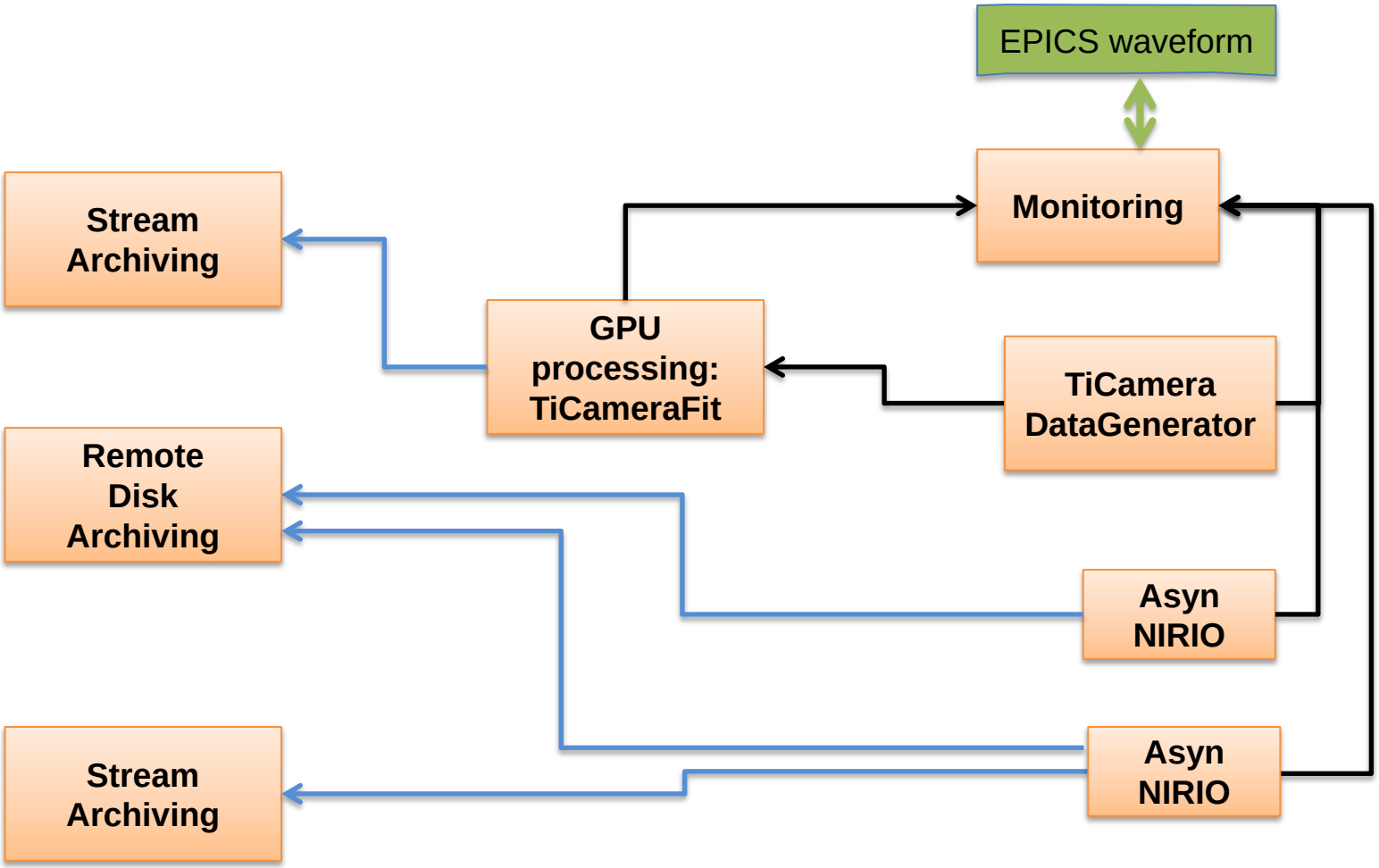
General Archiving Diagram



FPSC architecture



FPSC Internal data flow



Reasons for NetCDF-4

- Storage based on HDF-5
- Self Description Data
- Huge file sizes
- **Supports: single writer – multiple readers**
- Optimized direct data access to segments
- Data may be appended without copying or redefining its structure
- Previous netCDF versions are supported
- Complete utilities set
- Contributions and developments from wide scientific community

Reasons for NetCDF-4

- **Model for scientific data:** *variables, dimensions, attributes, coordinates*
- **Libraries for data access:** *C, Fortran, C++, Java, Perl, Python, Ruby, Matlab, IDL, ...*
- **Portable format: architecture independent**

- *Home page: http://www.unidata.ucar.edu/software/netcdf/*
  - *Clima and weather science community*

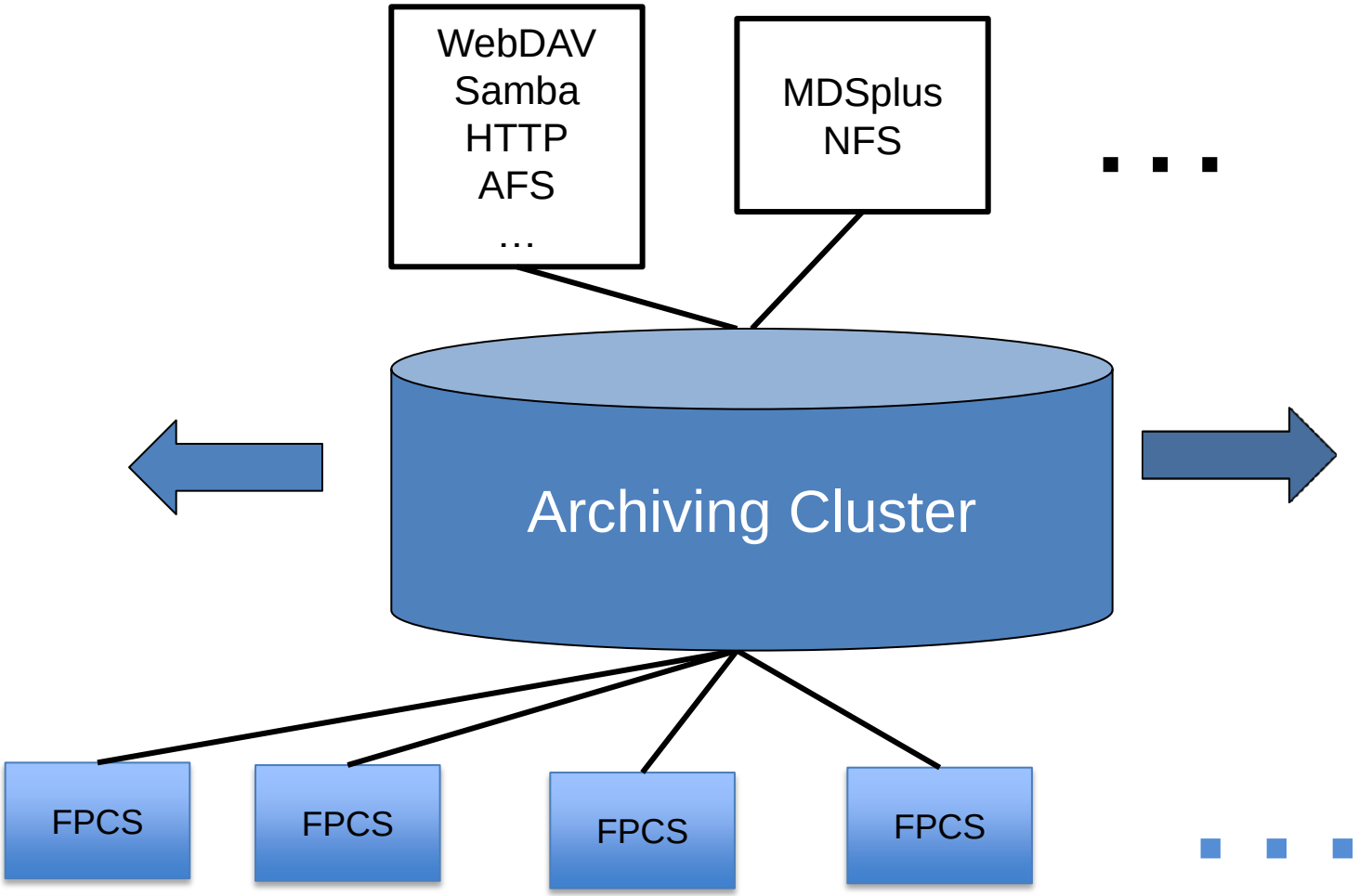
NetCDF “d1wave” type

```
netcdf fpsc_d1wave {
   types:
            uint(*) vlen_t;
   dimensions:
            acqtime = UNLIMITED ;
   variables:
            uint64  acqtime(acqtime) ;  // 64 bits (nanosecs) from the beginning of the file
                     acqtime:long_name = "Acquisition time" ;
            uint blocknumber(acqtime) ;
                     blocknumber:long_name = "Number of block";
            uint64 speriod(acqtime);
                     speriod:long_name="Sample period for this data block";
                     speriod:units="ns";
            uint64 srate(acqtime);
                     srate:long_name="Sample rate for this data block";
                     srate:units="samples/s";
            int nsamples(acqtime);
                     nsamples:long_name="Number of samples of this data block";
            vlen_t levels(acqtime);
                     levels:long_name = "Acquired values array" ;
   // global attributes:
            …
            :time_stamp_start_secs = 1000 ;
            :time_stamp_start_nanosecs = 10000000000 ;
            :scale_factor=1.0;
            :offset=0.0;
            :samplesize=0.0;
}
```

NetCDF “event” type

```
netcdf fpsc_event {
    dimensions:
            acqtime = UNLIMITED ;
    variables:
            uint64  acqtime(acqtime) ;  // 64 bits (nanosecs) from the beginning of the file
                    acqtime:long_name = "Acquisition time" ;
            uint blocknumber(acqtime) ;
                    blocknumber:long_name = "Number of block";
            // ----------------------------------------
            uint source(acqtime);
                    source:long_name="Source of the event";
            uint priority(acqtime);
                    priority:long_name="Priority level of the event";
            uint information(acqtime);
                    information:long_name="Event information code";
            string description(acqtime);
                    description:long_name="Description text of the event";
   // global attributes:
            :sourceID = "Channel_1" ;
            :pulseID = "345" ;
            :version = 1. ;
            :time_stamp_start_secs = 1000 ;
            :time_stamp_start_nanosecs = 10000000000 ;
}
```

Clustering Storage system



Archiving cluster requirements

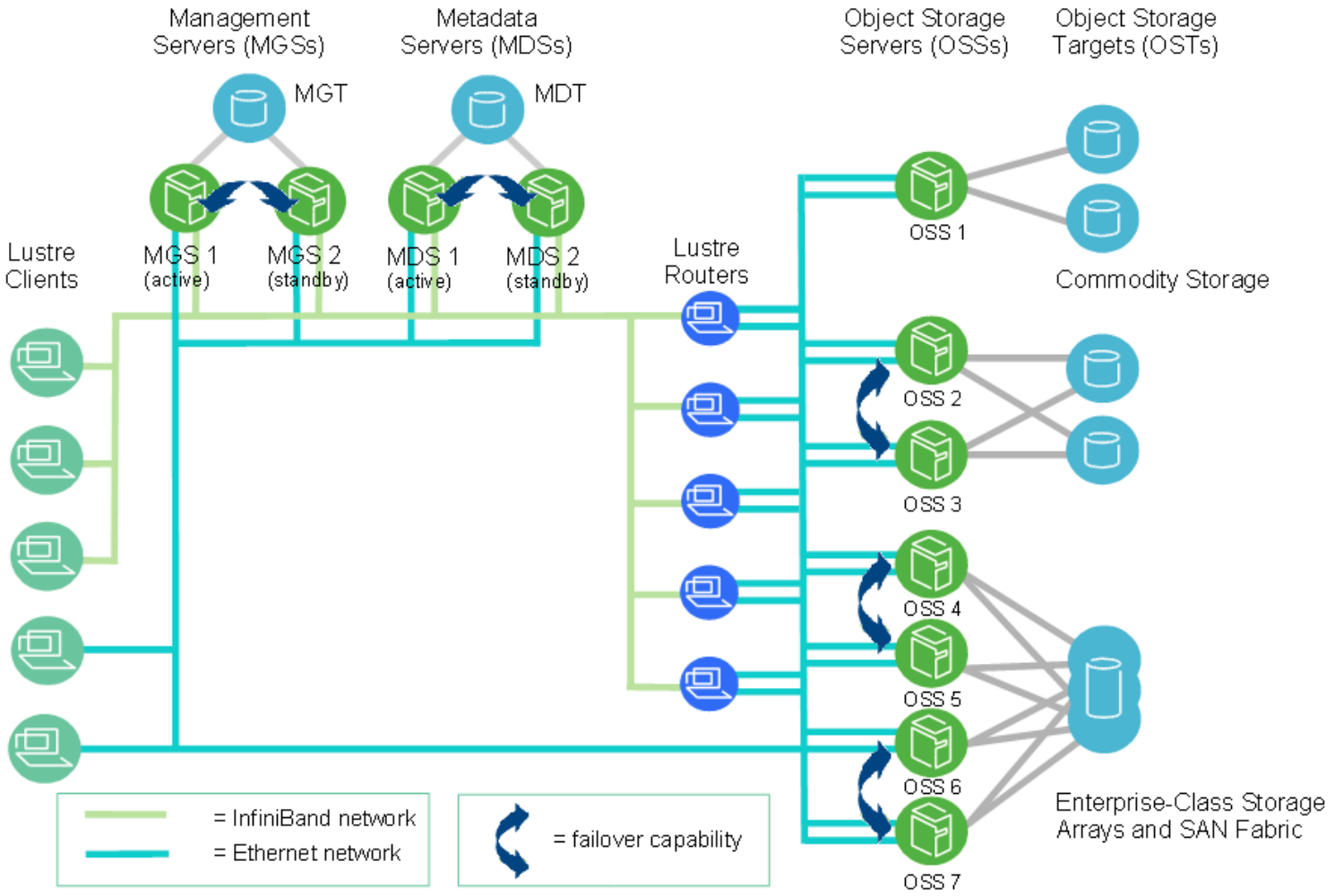
- Unique file system shared by all clients
- Fault tolerant (at least 1 server fault)
- Compatible with GEthernet connections
- Scalable
- Optimized for writing
- Concurrent (read / write)

- **Lustre**: Storage Clustering Solution -

Lustre clustering

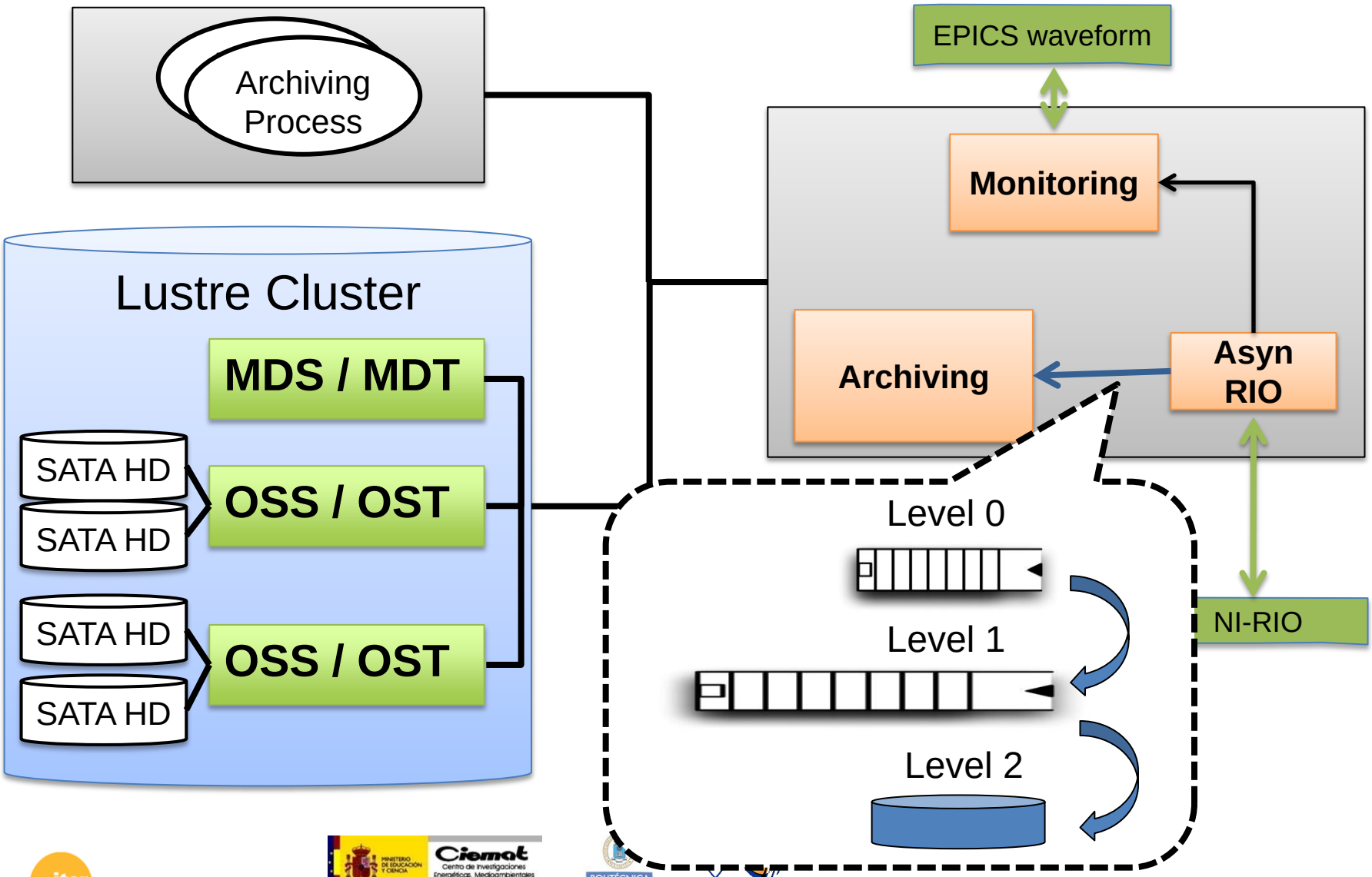
- All storage space mounted as a local file system
- Good scalable performance: limited by network or disks capabilities
  - File striping, up to 160 targets (OSTs)
  - Storage: 64 PB, file size 320 TB
  - Until 100,000 clients and 4000 OSTs
- Capacity to mix different network technologies
  - GEthernet, Infiniband, …
  - NFS is supported
- Fault tolerant capacity in all their components
- Good documentation and well supported
- Management tools

General Lustre Architecture



- Graphic obtained from Lustre Operations Manual -

The test installation

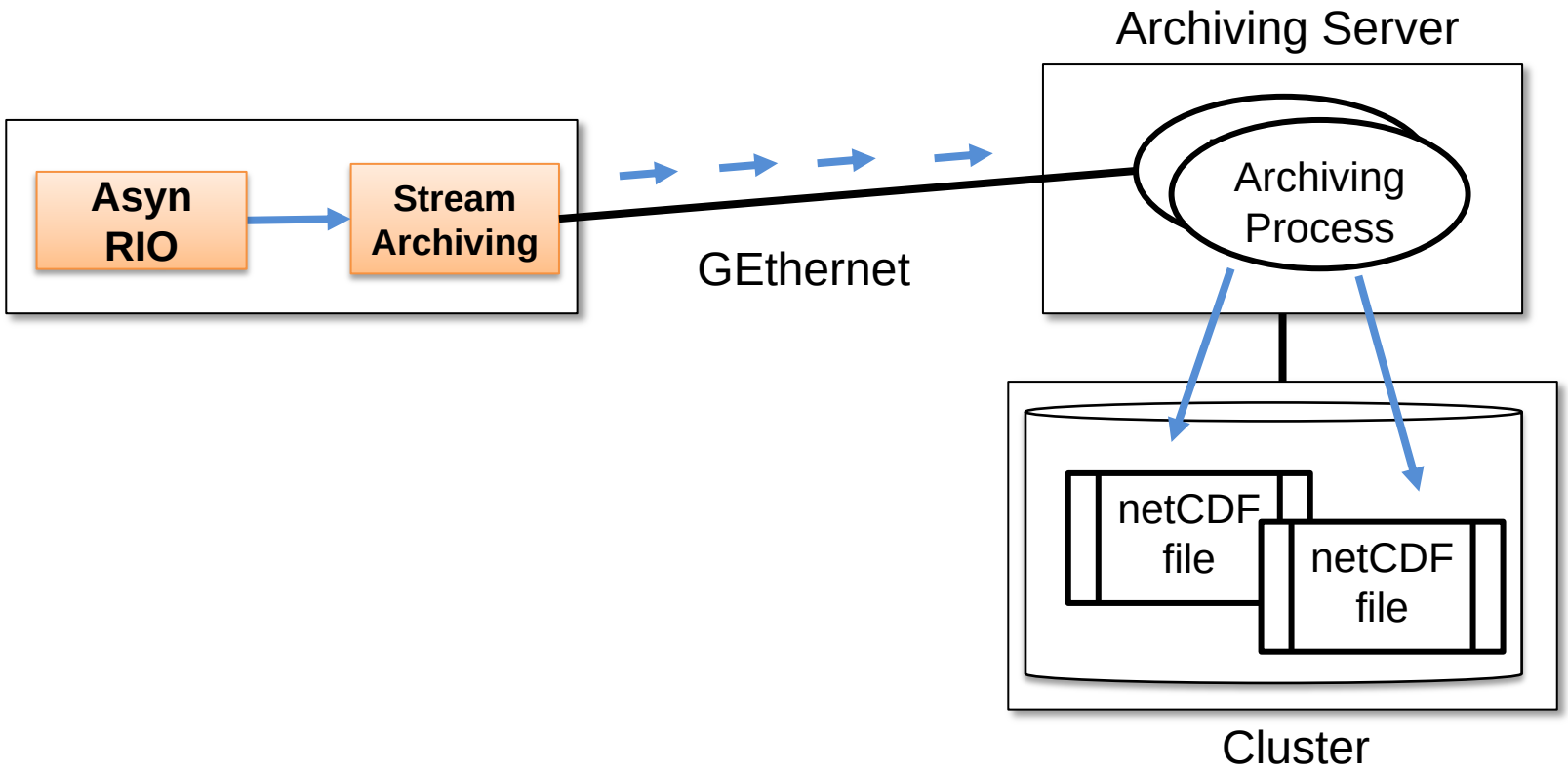


Installation details

- Cluster
  - MDS/MDT
    - 1 node CPU i5 650, 4GB RAM, 320 GB SATA HD
  - OSS/OST
    - 2 nodes CPU i5 650, 4GB RAM, 640 GB in 2 SATA HD RAID 0
- FPSC
  - CPU i3 540, 4GB RAM, H55 Intel Chipset
  - PXIe FlexRIO card connected with PCIe link to PC
- Remote Archiving Server
  - CPU i3 540, 4GB RAM, H55 Intel Chipset
- Network
  - 1GEthernet

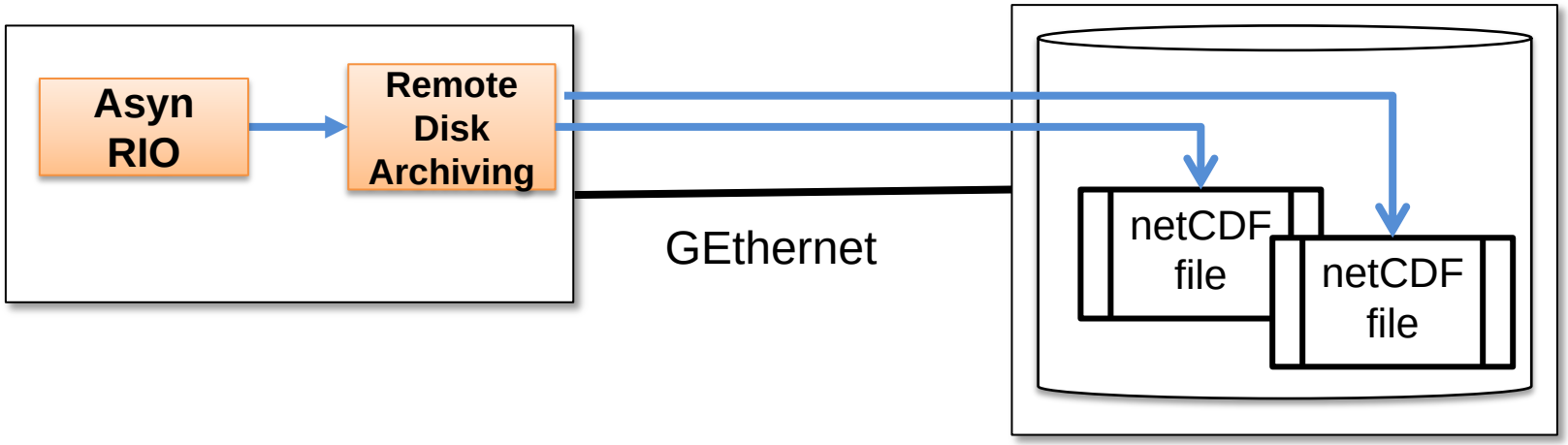
Test scenario 1

- To stream data to a remote archiving server that writes it in the cluster in NetCDF-4 format



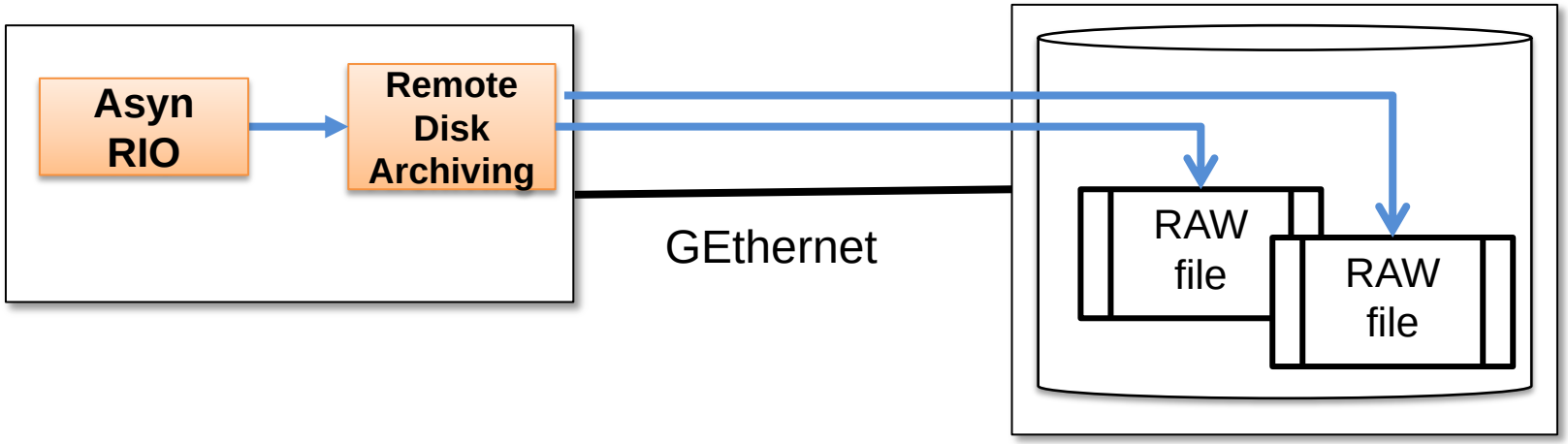
Test scenario 2

- To write in the cluster in NetCDF format
  - Cluster mounted as a local file system
  - Data written to disk using NetCDF-4 format



Test scenario 3

- To write in the cluster in RAW format
  - Cluster mounted as a local file system
  - Pulse and signal description header + data blocks directly written to disk



Results

- 1- To stream data to a remote archiving server
- 2- To write in the cluster in NetCDF format
- 3- To write in the cluster in RAW format

	Archiving Throughput	Av. CPU usage (before archiving)	Av. CPU usage (in archiving)
1st Scen.	40MB/s	20%	28%
2nd Scen.	24MB/s	20%	96%
3rd Scen.	98MB/s	20%	35%

- Using 3rd solution, we have been able to archive 24 channels at 1MHz during more than 1800 seconds
- Fault tolerance was successful achieve for Ethernet wire disconnection and reconnection during archiving

Conclusions

- Use of storing cluster as a local file system
  - Pros:
    - Complete fault tolerant solution (nodes, link, …)
    - Easy to maintain and scale in size and performance
    - Flexible (new archiving formats can be incorporated in a simple way)
    - Valid for different network types
  - Cons:
    - Only valid for Linux clients (valid for CODAC Core System v2)
    - FPSC CPU used on file formatting and archiving functions

Conclusions

- Use of NetCDF-4
  - Pros:
    - Architecture independent solution
    - Widely supported by many programming languages
    - Well supported
    - Easy to maintain data versions (self described format)
    - Easy to create formats for new data types
  - Cons:
    - High CPU demanding format for high rate data
    - Multi-threading not managed by HDF-5 libraries. It implements a walk-around.

Conclusions

- We have been able to archive 24 channels at 1MHz during more than 1800 seconds using RAW in cluster
- Fault tolerance was successful achieve for Ethernet wire disconnection and reconnection during archiving
- NetCDF-4 (HDF-5) saturates CPU for very high rate data archiving
- Cluster is a valid scalable solution in size and performance
- Cluster solves complex fault tolerant and management issues
- Some clients could require archiving server on remote
  - To avoid CPU saturation for complex data formatting
  - To do it compatible with storage cluster